

CSE 259 - Logic in Computer Science

Fall 2024

Recitation-1

Waqar Hassan Khan



Slides and Codes

- Slides and codes will be available in the Github repository - <https://github.com/Waqar-107/ASU-CSE-259-Prolog>

How to contact with me

- Email: wkhan17@asu.edu
- Discord handle: Waqar_107

I check both frequently!

Prolog - resources

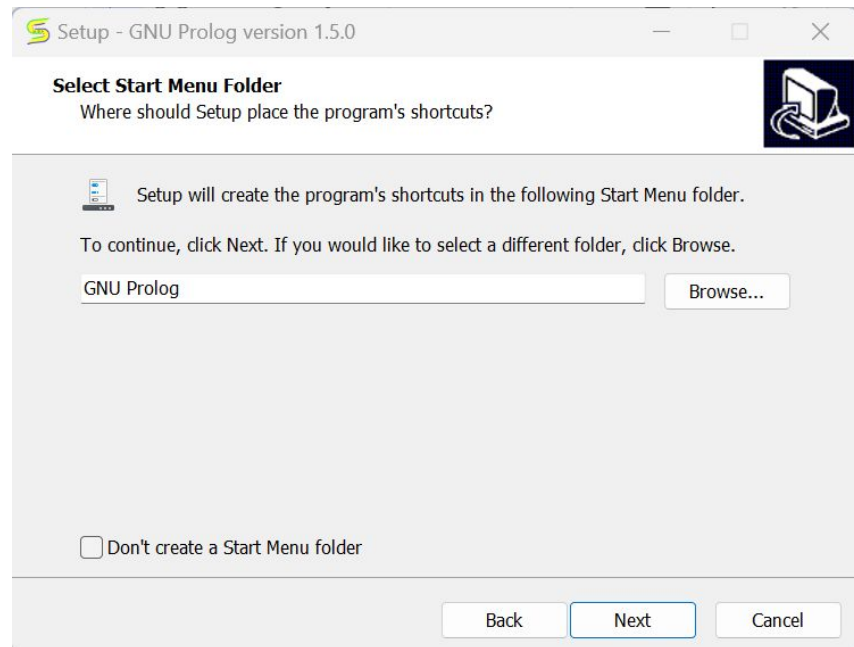
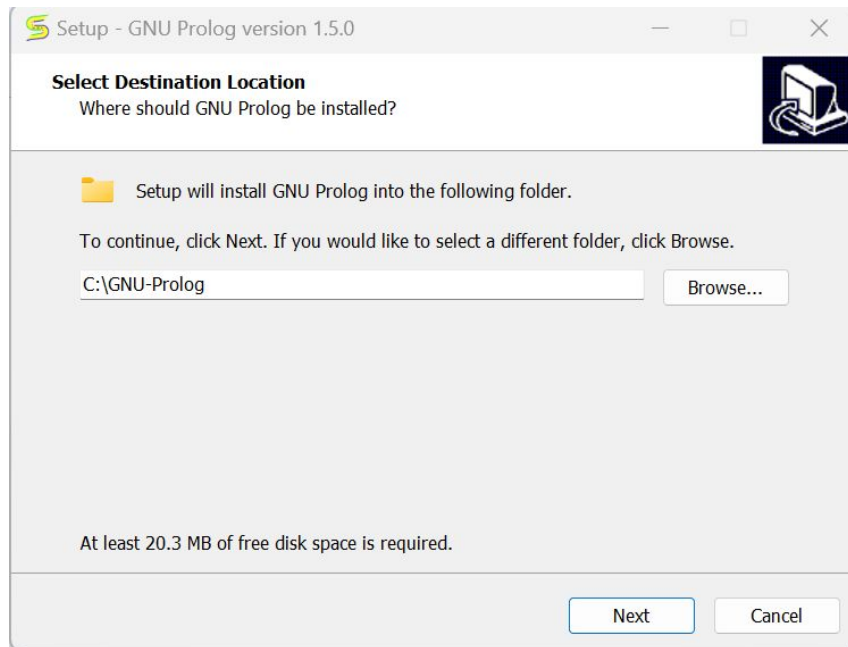
Tool we will be using:

- [GNU Prolog](#)

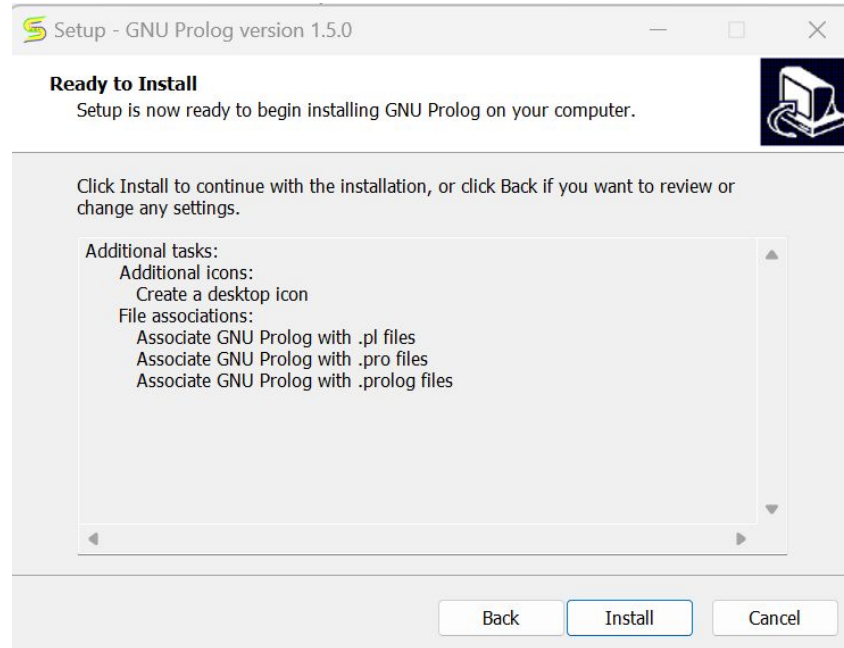
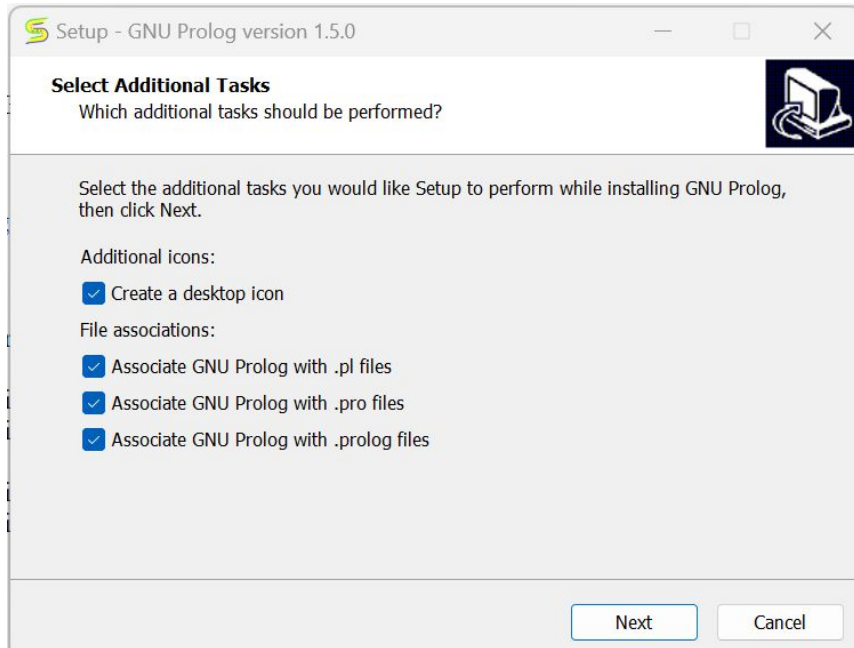
Useful resources:

- [GNU Prolog manual](#)
- [The Prolog Dictionary](#)
- [Prolog Tutorial - TutorialsPoint](#)
- [Prolog Tutorial from YouTube](#)

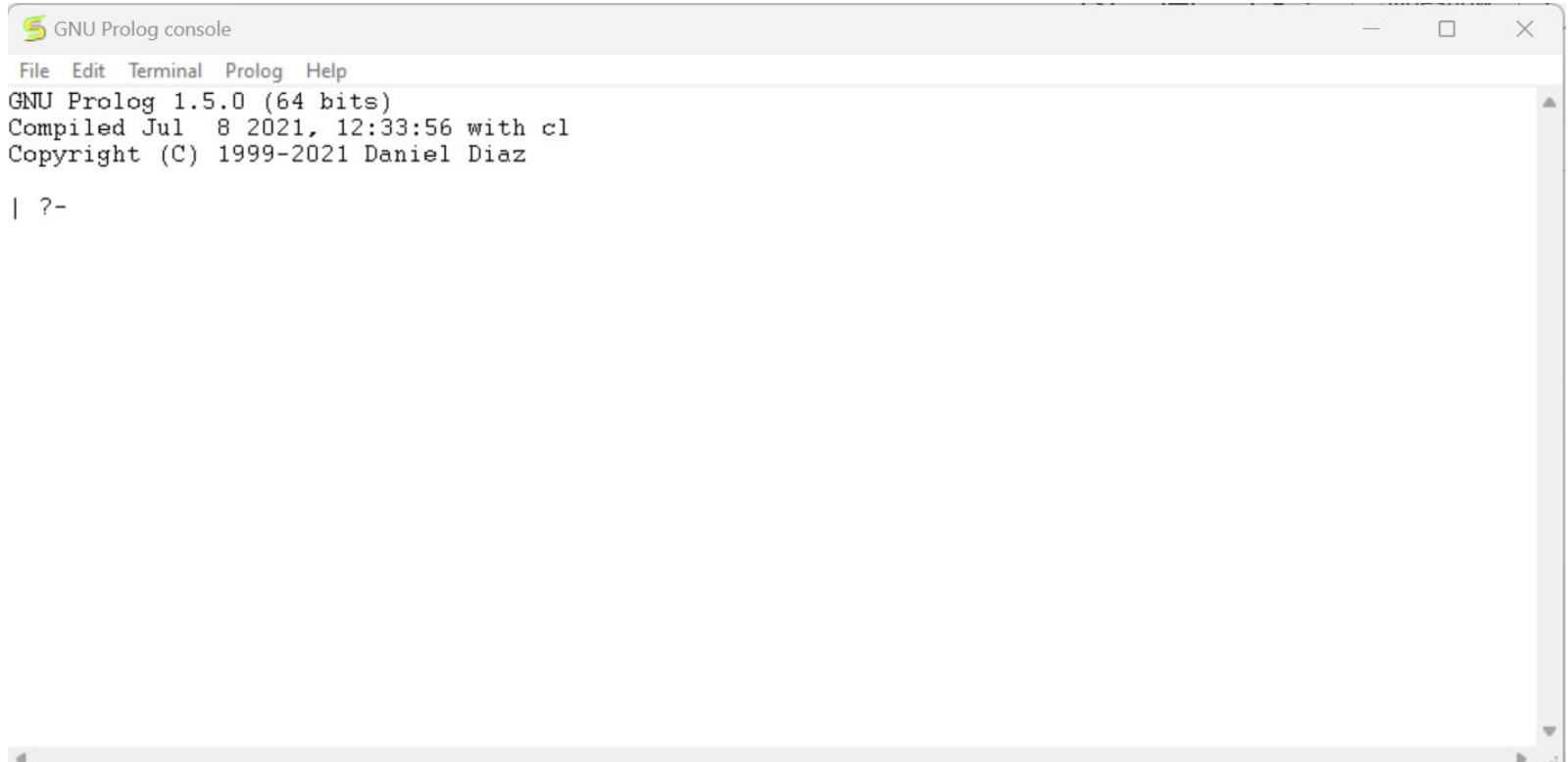
Installation - Windows



Installation - Windows contd.



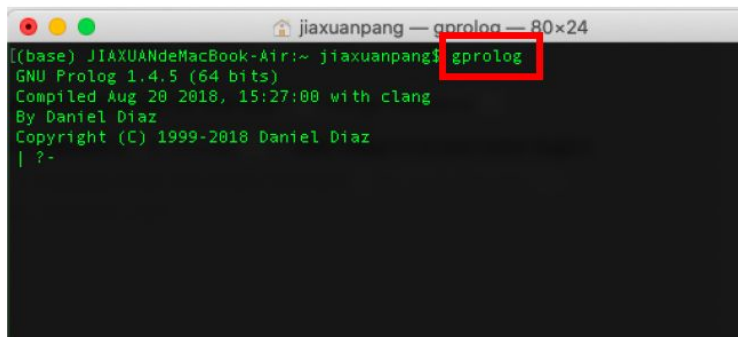
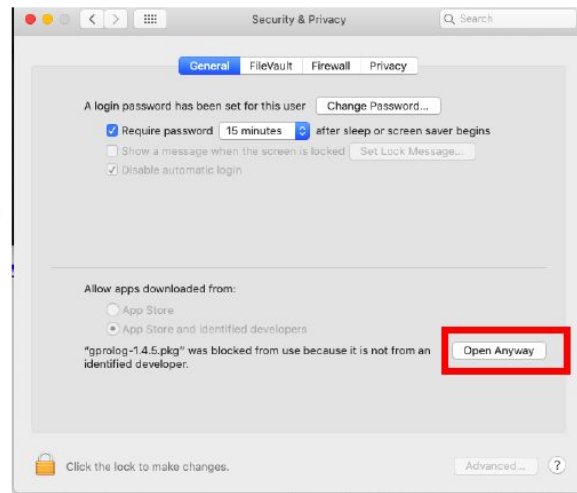
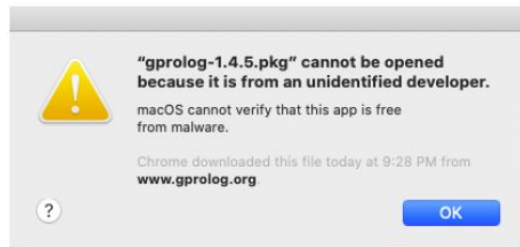
Installation - Windows contd.



The screenshot shows a window titled "GNU Prolog console" with a standard Windows-style title bar (minimize, maximize, close buttons). Inside the window, there is a menu bar with "File", "Edit", "Terminal", "Prolog", and "Help". The main text area displays the following information: "GNU Prolog 1.5.0 (64 bits)", "Compiled Jul 8 2021, 12:33:56 with cl", and "Copyright (C) 1999-2021 Daniel Diaz". The prompt "| ?-" is visible on the line following the copyright notice. A vertical scrollbar is present on the right side of the text area, and a horizontal scrollbar is at the bottom.

```
GNU Prolog console
File Edit Terminal Prolog Help
GNU Prolog 1.5.0 (64 bits)
Compiled Jul 8 2021, 12:33:56 with cl
Copyright (C) 1999-2021 Daniel Diaz
| ?-
```

Installation - MACOS



What is Prolog?

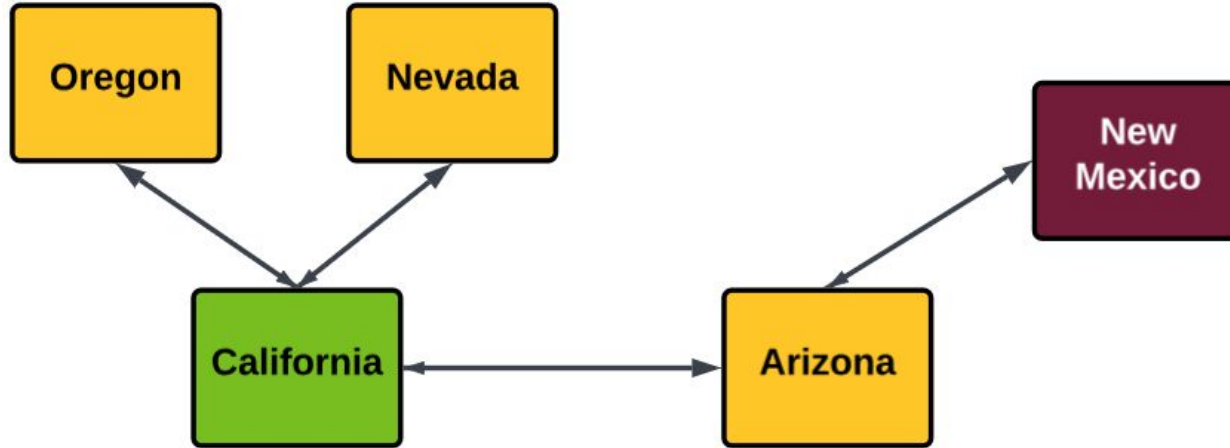
- Prolog is a **logic programming language**
- Prolog is intended primarily as a **declarative programming language**

Declarative Programming Language

Declarative programming is when you write your code in such a way that it describes what you want to do, and not how you want to do it. It is left up to the compiler to figure out the how.

Example: SQL, Prolog (Obviously :D)

An example



Suppose, we are in California. Which states can we visit from here? (We can travel if the states are adjacent)

Sample code for the example - Java

```
1 public class StateTravel {
2
3     public static void main(String[] args) {
4         boolean oregonConnected = true;
5         boolean nevadaConnected = true;
6         boolean arizonaConnected = true;
7         boolean newMexicoConnected = false; // Arizona only connects to New Mexico
8
9         // Individual if cases for each state
10        if (oregonConnected) {
11            System.out.println("Can travel to Oregon from California.");
12        } else {
13            System.out.println("Cannot travel to Oregon from California.");
14        }
15
16        if (nevadaConnected) {
17            System.out.println("Can travel to Nevada from California.");
18        } else {
19            System.out.println("Cannot travel to Nevada from California.");
20        }
21
22        if (arizonaConnected) {
23            System.out.println("Can travel to Arizona from California.");
24        } else {
25            System.out.println("Cannot travel to Arizona from California.");
26        }
27
28        if (arizonaConnected && newMexicoConnected) {
29            System.out.println("Can travel to New Mexico from California (via Arizona).");
30        } else {
31            System.out.println("Cannot travel to New Mexico from California.");
32        }
33    }
34 }
35
```

Sample code for the example - Prolog

```

1      next_to(oregon, california).
2      next_to(california, oregon).
3
4      next_to(california, nevada).
5      next_to(nevada, california).
6
7      next_to(california, arizona).
8      next_to(arizona, california).
9
10     next_to(arizona, new_mexico).
11     next_to(new_mexico, arizona).
12
13     travel(A, C) :- (next_to(A, C) ; (next_to(A, B), next_to(B, C), A \= C)).

```